From “Gigahurts” to Gigahertz – The Process of Silicon Debug

Doug Josephson, Hewlett Packard Company

ddj@hp.com

Introduction

My cell phone rings. When I answer the call, I hear four words - “The shmoo is clean!” Unsure that I have heard the words correctly, I ask the caller to repeat the message, and after he does, I stand up, and let out a loud, extended “YES!!!” and the sound reverberates through the empty forest of cubicles around me. With this call, another victory has been won in the endless fight against silicon design bugs. A last minute, showstopper bug has been discovered, debugged and (as the shmoo showed) fixed in a matter of days, and the product can ship on time. Debugging the failure and resolving it required utilization of numerous design features for debug, many debug experiments, several failure analysis tools, and sheer engineering ingenuity.

Experiences such as this are the province of the silicon debug engineer. Debug engineers must have multi-disciplinary skills, spanning areas such as transistor-level physical design, computer architecture, failure analysis tools, software, packaging, system integration, silicon processing, and failure analysis. They must couple the ability to integrate and apply knowledge in each of these areas, along with teamwork and problem solving skills, to discover, debug and fix silicon design issues before they are ever seen by a customer. Silicon debug can be one of the most frustrating, yet rewarding jobs an engineer can perform.

Silicon Characterization and Debug

Silicon characterization and debug begins once the initial design is completed and manufactured. Characterization consists of validating that the design does what it is supposed to do. Debug is the process of determining the cause of failures found during characterization. Once design failures are "root caused" through the process of debug, fixes are implemented and verified after they have been applied to the design.[1-3]

Characterization begins on automated test equipment, where initial test vectors are applied to the device to determine good chips from bad ones. Once good devices are found they are packaged, retested to find any problems introduced through the assembly process, and assembled into the target system they were designed for to allow additional testing. Design problems are identified either on a tester or in a system environment during the process of characterization. Failures on a tester may show up as early/late/missing/incorrect patterns at the pins. System failures can manifest themselves as failures during boot-up of the system, during focused or randomized system testing, or even in system applications.

Both testers and systems are useful for verification and debug, and each offer advantages and disadvantages. Testers offer excellent control over parameters such as voltage, frequency, and pin timing, and operate in a deterministic way that can ease debug. They are also fast and can characterize many devices far more quickly than can be done in a system. However, testers are also very expensive and cannot comprehensively test the device with as many vectors as the system can. In contrast, systems are usually far cheaper than testers, and have considerably fewer limitations in applying diverse and extensive test patterns to the device. However, it is more difficult to control environmental parameters such as voltage, temperature, and frequency in a system, and device behavior in a system can be more difficult to debug than on a tester due to non-determinism that sometimes exists in system environments. As a result of the complementary nature of testers and systems, both are commonly used in silicon characterization and debug.

Characterization and debug is split into two areas: functional and electrical. Functional characterization tests the device for logical correctness – does it produce the correct results at a nominal operating point? Errors in functionality are due to incorrect logical design. Electrical characterization testing verifies that the device functions correctly at all specified operating points. Electrical characterization can be much trickier than functional, since it is possible for a device to operate correctly under some conditions but not others. Electrical characterization is focused on guaranteeing the robustness of the device across voltage, frequency, temperature, and silicon processing variations. It is useful to vary all of these parameters during characterization to expose electrical design marginalities.

Once failures are identified during the characterization process, the process of debug begins. Failures are typically triaged into similar failure modes. The worst test case that causes the failure is found, and the problem is handed off to a debug team. Debug teams may work in parallel on different failures. They perform experiments to determine sensitivities that a particular bug has (e.g., varying temperature, performing test vector experiments, looking at pin values, trying different parts with different processing, etc.)

Functional debug teams examine the vectors applied to the device, and try different experiments to change the ordering of events to the device. Logic analyzers are useful in determining the response of the device. Traces of activity at the pins may be useful to the debugger in determining why the chip does not operate correctly. It may be possible to simulate the failure in the pre-silicon RTL simulation environment by extracting the internal state of the device and using information gathered on a logic analyzer. This allows more observability into what is happening during the failure and possibly indicating the root cause for the problem.

One of the most important techniques for electrical debug teams is the "shmoo".[4] Shmoos can be performed in either systems or on testers. Typically, frequency and voltage are varied to expose overall sensitivities in the design. Shmoos run at hot and cold temperatures can also give great insight into failures. For example, shmoos that show failures at high temperature, but not cold temperatures may indicate problems with

leakage (since FET/diode leakage increases), noise (since threshold voltages decrease, making noise-sensitive circuits easier to spuriously trip), or speed (since FETs slow down). Shmoos that fail at cold temperatures, but not at high temperatures, may indicate problems with races (since FETs and thus circuits speed up), noise (since edge rates decrease and dynamic currents increase, thus increasing di/dt and dv/dt for inductive and capacitive coupling), or charge sharing (since FETs speed up and charge is more rapidly transferred between sensitive nodes).

There are numerous different characteristic shmoos that are typically encountered during silicon debug. These often vary considerably from the traditional "speedpath" shmoo encountered in CMOS designs (increasingly fast operation with increasing voltage). Debug engineers often assign euphemistic names to each characteristic shmoo during debug that describe the relative characteristics of each shmoo. Figure 1 shows some examples of commonly encountered shmoo plots.

Other techniques external to the chip are also used by debug engineers. These include tools such as logic analyzers that allow the debugger to trace down when incorrect or missing results occur at the pins of a device, and oscilloscopes that can be used to debug analog/timing related events on external pins of the device. However, with the ever-increasing density and complexity of designs, one of the most important ways to ensure that rapid and productive debug is possible after silicon is received is to ensure that hardware features to support debug are present in the design before it is even released for initial manufacturing. Such features are the most important tools in debugging failures.

Design for Debug – "Did You Bring the Lifejackets?"

Setting sail on the sea of silicon debug can be a dangerous journey unless one is prepared. Even when the ship is guaranteed "unsinkable" by the design team, the prudent debug engineer ensures that the lifejackets and lifeboats are on board! Most chip designs today support some sort of debugging hardware built into the design to facilitate characterization and debug.[5-12] Often, many of the design features added to support manufacturing testability are reused to support debug as well. One key example of such

reuse is scan techniques that can observe internal states of the device. For testability reasons, many devices include scan capability that allows the manipulation of internal state elements (latches/flip flops) in the design by connecting them together in one or more "scanpaths". Scanpaths are serially accessible external to the device, usually using the IEEE 1149.1 protocol for access. This allows easier testing of a device by partitioning it into smaller pieces of logic; internal signals can be controlled and observed for testing in this way. Since many internal signals can be observed with scan, it is also very useful to enable debug through the examination of internal signals of the device during both failing and passing conditions.

Two types of scan are typically employed for debug – *destructive* and *non-destructive*. Depending on the scan implementation chosen, it may be required to stop the device's clock(s) to enable scanning of internal state to occur (destructive scan). However, an alternative is to use non-destructive scan that allows single cycle observability of internal signal state while the chip continues to operate. Designs may use a mix of both types, with non-destructive scan allowing the debug engineer to pinpoint an area of failure to a small region, whereupon more extensive destructive scan can be used to observe more signals. The ratio of destructive to non-destructive scan in a design might be 10:1 or more (because non-destructive scan requires more silicon area than destructive). The advantage of non-destructive scan is that it is non-invasive to chip logical state – samples of signals may be taken without altering the clocking of the device. This can be useful when debugging failures in the system that a chip operates in – stopping the clock to take one sample at a time can be extremely slow and inefficient.

In order to sample signals at the appropriate time, some sort of internal triggering functionality is required. This triggering circuitry observes many critical signals on the chip (e.g., addresses, instructions, data, and control signals) and allows programmable trigger signals to be generated through the use of pattern matches and mask bits. Simple trigger signals can be used to create complex trigger combinations. For example, a debug engineer may set a trigger to recognize the third occurrence of a particular instruction in a loop, then wait 12 clock cycles, and look for a data pattern on a data bus. If the pattern occurs, a trigger signal is generated to take a scan sample of the internal state that can

then be scanned out and examined/compared between a passing and failing case. Such complex triggering can also be combined with large memories present on some chips to enable on-chip logic analyzer capability. By setting aside a portion or all of a memory unit, internal signals can be captured sequentially based upon an initial and final trigger to allow for later observation, or can even be propagated out of the chip via special debug pin interfaces.[13]

Clock manipulation is another debug feature for which triggering is important.[1, 5, 11-12] Since most chip designs use a synchronous clock, the ability to control the clock waveform for the design can be a useful debug feature. Changing duty cycle, clock period, and clock phase length is extremely helpful in identifying the "critical" cycle/phase where a failure occurs. When coupled with complex triggering capability, clock manipulation can be algorithmically applied across an entire test to determine if any "sensitive" cycles occur in the test. One by one, each cycle can be manipulated to see what effect it has on making a passing test fail, or a failing test pass. Typically clocks can be "stretched" and "shrunk" to enable this. For example, if a failure is observed for a particular test sequence, the test can be set to run at a "safe" or passing frequency near the frequency of failure. One by one, each clock cycle/phase in the vicinity of the failure can be "shrunk" to see if doing so causes the failure to occur. Alternatively, the frequency can be set to a known "unsafe" frequency where the test fails, and each cycle/phase can be "stretched" to see if the failure goes away. In addition to manipulating the waveform, digital manipulations are also sometimes used to "drop" a clock on a specific cycle. This can be useful in debugging silicon with multiple clock domains that are asynchronous with respect to one another. Examples of such analog and digital clock manipulations are shown in Fig. 2.

By performing such experiments, it is possible to narrow down the failure to a particular clock cycle in time, which can be enormously useful in pinpointing the failing circuit. Armed with this information and knowledge of what operations are in progress in the device at that time, it is usually possible to theorize why a particular circuit failure might be occurring.

Yet another useful debug feature in the clock system is programmable clock skewing.[1] Typically, designs use one or more logical clocks. In order to distribute such clocks across the device, many physically different clock buffers may be used. It is normally desirable to minimize the clock "skew," or difference in time, between each of these buffers, so that they all generate the reference clock at the same time for each circuit in the device. This simplifies design and timing analysis by providing identically timed clock edges across the device. However, it can be advantageous in debug to introduce intentional skew between the clock buffers. The skewing of clocks generated by separate buffers in different regions can give a speedpath between two circuits clocked by different buffers more time to evaluate, or a race may be avoided between two circuits by adjusting their clocks. If this capability is provided at a fine enough level, it can even be possible to set many buffers to slightly different times to maximize the frequency of operation of various speedpaths. An example of clock skewing is shown in Fig. 3.

Finally, physical debug is another area where design hooks are used. To support various failure analysis and diagnostic tools, it is essential to provide both navigation and access features in the design. Navigation fiducials are often placed in a design, particularly in flip-chip designs (where the front side of the chip is "flipped" over and connected with solder balls to a package or interposer), to aid in navigation from the backside of the silicon via infrared imaging. Fiducials can be passive or active structures, e.g., metal lines arrayed in a particular manner or diodes that emit infrared light. Design access features and rules are also important. These can include diffusion/metal placed specifically to enable probing techniques, spare gates placed to enable metal-only edits, and rules to ensure accessibility to wiring and gates to allow probing and edits of the design. These features, when combined, allow the use of various failure analysis tools and methods.

FA Tools and Techniques To the Rescue!

Fortunately, due to wiles and hard work of many engineers in the FA world, there are many opportunities to save the unwary designer from himself and end up looking like heroes in the process! Many tools familiar to the failure analysis engineer that are often

used for silicon process debug and defect analysis are also used in the process of silicon design debug. When combined with the information gleaned from shmoos, vector experiments, and external/internal debug features, they are a final step in establishing root cause of design failures. In addition, they can be extremely useful in perturbing particular failures, and also in trying out various possible design fixes prior to actually making silicon design changes. Such capability is vital in reducing the likelihood of a "bad" design fix that does not completely address the failure, as well as the concomitant impact to project schedule.

Several tools can be used to collect information about failures. Three commonly used data acquisition tools are e-beam probing, picosecond imaging circuit analysis (PICA), also known as time-resolved emission (TRE)[14-16], and laser voltage probing (LVP)[17]. E-beam probing uses an electron beam and secondary emission of electrons to probe voltages on internal nodes of an integrated circuit. A major drawback of e-beam probing is that the part must be debugged in a vacuum; this can lead to problems in removing heat from high-power devices. PICA depends on emission of infrared photons generated by hot carriers within CMOS devices in saturation to monitor circuit activity. It is an extension of emission microscopy (EMMI) that allows time as well as spatial resolution of emission intensity from hot carriers. LVP uses an infrared laser beam to probe voltages on internal nodes via the Franz-Keldysh effect. Both PICA and LVP are techniques that can be used through the backside of silicon devices since both use infrared wavelengths, and silicon is transparent to these wavelengths. Parts must be thinned prior to PICA or LVP to reduce infrared absorption. e-beam probing can resolve signals on metal lines, whereas LVP requires diffusion for probing, and PICA requires a switching transistor. Both e-beam probing and LVP can be invasive to the circuit being measured, while PICA is non-invasive since it passively measures emission only. All three techniques must repeatedly generate a trigger synchronized to the failure within the device. This is accomplished using the internal debug trigger circuitry (described previously) of the IC being measured to toggle an external signal pin that triggers the probing system.

All of these tools can be useful in characterizing a particular failure. However, it is usually necessary to have a reasonably good guess as to where a particular failure is occurring in the device prior to using one of these tools – it is difficult to debug a failure otherwise. An analogy would be using a flashlight in a dark auditorium to locate a particular seat without a good idea of where the seat is beforehand – one can look for a long time and may never find the seat! Narrowing in on the failure to enable these probing techniques is typically done by analyzing passing and failing scan information, or through experiments with test vectors that indicate where the failing circuit might be. Additionally, failing clock cycle information is needed to ensure that the trigger is generated at the right time to capture the passing and failing waveforms or images. Another critical need is the ability to precisely navigate to the correct nodes to probe, which is enabled through the previously mentioned navigation fiducials.

Internal nodes can be probed in the device by looping over a failure and generating a regular trigger signal from the device being measured that then triggers the measurement system. LVP and e-beam can build up a waveform similar to an oscilloscope waveform of internal nodes with timing resolution of approximately 50 ps. PICA images can indicate the relative arrival of signals in an area, and the emission intensity can be useful in diagnosing events such as local voltage depression or the relative voltage between circuits.

Armed with information collected from shmoos, scan dumps, clock manipulation, and tools such as PICA and LVP, the debug engineer can formulate a theory as to why a particular circuit is failing, and additional experiments can be performed to verify the theory. SPICE circuit simulation can also be used to explore possible failure scenarios based upon the information gathered from shmoos, scan, and FA tools. Root cause is achieved once the "light switch" is discovered – the mechanism by which the failure can be made to appear and disappear.

One way to gain confidence in the theory is to perform a focused ion beam (FIB) edit to the failing circuit to either fix the predicted failure, or to perturb it in some way.[18-19] For example, if a clock is late in arriving in a particular circuit, it may be helpful to cut away

some delay in the local clock buffer for a circuit to speed the clock up and see if the failure goes away. Conversely, a FIB edit can add capacitance to a node by adding additional metal to slow a signal down. FIB edits can also be combined with previously placed spare gates in the design to enable new logic to be formed to fix bugs as well. This can be essential in testing out the proposed fix in the design prior to having to make new masks, fabricate the design and test it, which can take a prohibitively long time. A FIB can be done in a matter of hours, and the corrected part can then be tested in a system and on a tester to validate the proposed fix. This is typically the final step in verifying that root cause of the failure is completely understood, and that the proposed fix will completely eliminate the problem in a future revision.

Future Challenges for Silicon Debug

As designs grow in complexity, so does the process of silicon debug. The voracious appetite of design teams for additional transistors, trickier circuits and more speed continues to be satiated by the "culinary heroics" of the mighty fab engineers as they cook up faster and better processes. This can only lead to one thing for the hapless debug engineer – indigestion! While great strides have been made in the process of silicon characterization, on-chip debug hardware, and FA tools, numerous challenges lie ahead.

One of the most pressing issues is simply power. As fab processes are pushed to their limits, transistors become leakier, with even gate leakage becoming a dominant factor. Leakage power can contribute 30% of total overall power in a 90 nm process. In addition, higher clock rates continue to push power upwards as well. Thermal density and the ability to remove heat from the device, particularly in debug situations where devices are deprocessed to facilitate FA tool use, are becoming difficult issues to solve.

Designs must change to be more power-aware in the future if they are to continue to advance in performance. Since $P = CV^2F$ for dynamic power, it can only be modulated through reduction in capacitance, voltage, or frequency. Voltage scaling is reaching its limits, and frequency continues its ever upward trend. Designs of the future will need to dynamically adapt themselves to different voltage and frequency environments to control

power, which will create difficult problems for debugging. Devices that change frequency and voltage lead to asynchronous, unrepeatable behavior. This behavior can become a nightermare to debug.

Another problem is external interface speeds. Automated test equipment is barely able to keep up with today's high pin count, high frequency devices. ATE will have to undergo a transition in the future to support more on-chip DFT features for testing, as it will simply become economically impractical to support high-speed test with accurate parametric control. Such issues may lead to more reliance on custom test solutions and DFT embedded within devices that allow them to be tested/debugged in the systems that they were designed for.

Reliability of circuit designs over process variation in leading edge processes is a growing concern. Variation in transistor performance across even a single die can cause robustness issues if not carefully planned for. Thermal variation in clock networks across a die must be taken into account to lower clock skew to acceptable levels. The ability to actively "tune" clock networks for such variation will be needed to support future debug. The ability to internally determine signal timing through debug hardware (e.g. on-chip timing analyzers or even oscilloscope-like functionality) will become increasingly important for debugging.

Another area for debug hardware innovation is in the active injection of various errors in hardware to explore how the design reacts to failures that may be infrequent in real application and thus very hard to validate during debug. This may expose design mistakes that might not otherwise be visible without a specific set of conditions. Additional work is needed in tools to support the identification of "sensitive" areas in designs and to target testing at these areas. This is similar to the process of ATPG (automated test pattern generation), but is more difficult since it must consider aspects of parameters such as signal timing, voltage variation, noise margin in circuits, etc., to determine how to expose weaknesses in the design.

Conclusion

The pace of integrated circuit design innovation is truly astonishing. In the past thirty years, circuit designs have grown from a trivial thousand transistors clocked at a measly megahertz, to near-future designs of over a billion transistors at frequencies over a gigahertz. Somehow debug capability has managed to keep pace with this staggering growth. Perhaps within another thirty years, we will see the advent of self-diagnosing and self-repairing designs. Not unlike one human "debugging and repairing" another today, perhaps these will perform the work of today's debug engineer, and we will wonder how we ever were able to get along without such capability. But at least for the near future, the job security of the debug engineer seems secure!

References

[1] D. Josephson, "The Manic Depression of Microprocessor Debug," *Proceedings of the International Test Conference,* 2002, pp. 657-663.

[2] J. Bockhaus, et al., "Electrical Verification of the HP PA8000 Processor," *Hewlett Packard Journal,* August 1997, pp. 32-39.

http://www.hpl.hp.com/hpjournal/97aug/aug97a4.pdf (accessed 4/6/2003)

[3] M. Bass, et al., "Design Methodologies for the PA7100LC Microprocessor," *Hewlett Packard Journal,* April 1995, pp. 6-7, 10-13.

http://www.hpl.hp.com/hpjournal/95apr/apr95a3.pdf (accessed 4/6/2003)

[4] K. Baker and J. V. Beers, "Shmoo Plotting: The Black Art of IC Testing," *IEEE Design and Test of Computers*, July/September 1997, Volume 14, Number 3, pp. 90-97.

[5] D. Josephson, et al., "Debug Methodology for the McKinley Processor," *Proceedings of the International Test Conference,* 2001, pp. 451-460.

[6] H. Balachandran, et al., "Facilitating Rapid First Silicon Debug," *Proceedings of the International Test Conference,* 2002, pp. 628-637.

[7] B. Vermeulen, et al., "Core-based Scan Architecture for Silicon Debug," *Proceedings of the International Test Conference,* 2002, pp. 638-647.

[8] X. Gu, et al., "Reusing DFT Logic for Functional and Silicon Debugging Test," *Proceedings of the International Test Conference,* 2002, pp. 648-656.

[9] C. Pyron, et al., "Silicon Symptoms to Solutions: Applying Design-for-debug Techniques," *Proceedings of the International Test Conference,* 2002, pp. 664-672.

[10] A. Carbine and D. Feltham, "Pentium(R) Pro Processor Design for Test and Debug," *Proceedings of the International Test Conference,* 1997, pp. 298-299.

[11] A. Kinra, et al., "Diagnostic Techniques for the UltraSPARC(TM) Microprocessors," *Proceedings of the International Test Conference,* 1998, pp. 480-486.

[12] T. Wood, "The Test and Debug Features of the AMD-K7(TM) Microprocessor," *Proceedings of the International Test Conference,* 1999, pp. 130-136.

[13] T. Litt, "Support for Debugging in the Alpha 21364 Microprocessor," *Proceedings of the International Test Conference,* 2002, pp. 584-589.

[14] J. Tsang, et al., "Picosecond imaging circuit analysis," *IBM Journal of Research and Development,* Vol. 44, Nos. 4, 2000, pp. 583-603.

[15] W. Huott, et al., "The Attack of the 'Holey Shmoos': A Case Study of Advanced DFD and Picosecond Imaging Analysis (PICA)," *Proceedings of the International Test Conference,* 1999, pp. 883-891.

[16] D. Knebel, et. al., "Diagnosis and Characterization of Timing-Related Defects by Time-Dependent Light Emission," *Proceedings of the International Test Conference,* 1998, pp. 733-739.

[17] M. Paniccia, et al., "Novel Optical Probing Techniques for Flip Chip Packaged Microprocessors," *Proceedings of the International Test Conference,* 1998, pp. 740-747.

[18] R. Livengood, et al., "Advanced Micro-Surgery Techniques and Material Parasitics for Debug of Flip-Chip Microprocessor," *Proceedings of the International Symposium for Testing and Failure Analysis,* 1999.

[19] Y. Hong and M. We, "The application of novel Failure Analysis Techniques for Advanced Multi-layered CMOS devices," *Proceedings of the International Test Conference,* 1997, pp. 304-309.

Contact Information

Doug Josephson

Hewlett Packard Company

3404 East Harmony Road, MS55

Fort Collins, CO 80528

ddj@hp.com

Biography

The author is a master engineer with Hewlett Packard in Fort Collins, Colorado. He received a BSEE from the University of Iowa in 1988. He has worked on the design, test and debug of high performance microprocessors for 14 years in engineering and management roles. He holds 6 patents, with several pending, has written several papers on design, debug and testing, and received the honorable mention award for the paper "Debug Methodology for the McKinley Processor" at the 2001 International Test Conference.

CAPTIONS

Fig. 1 Examples of commonly encountered shmoo plots.

Fig. 2 Examples of analog and digital clock manipulations

Fig. 3 Example of clock skewing

Figures

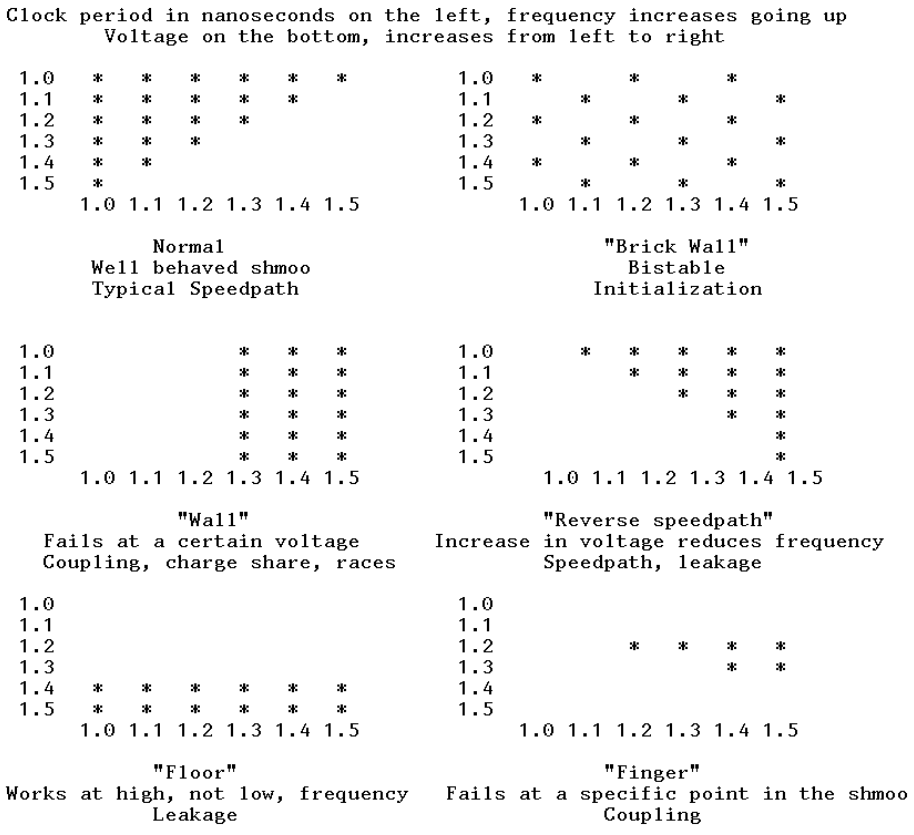
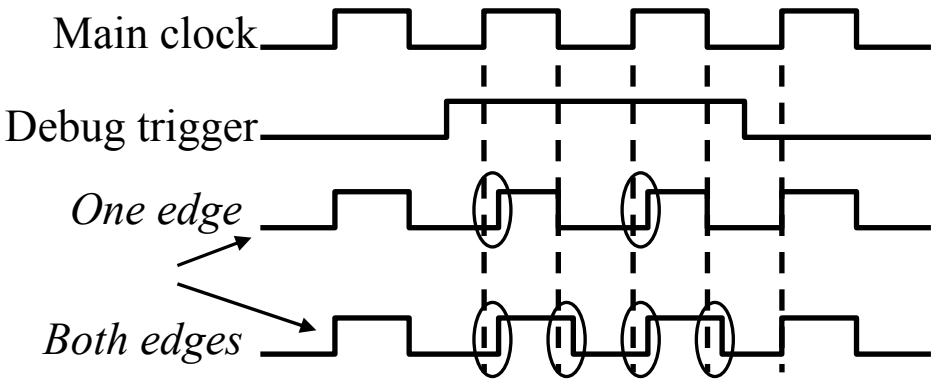


Figure 1

Analog Clock Manipulation



Digital Clock Manipulation

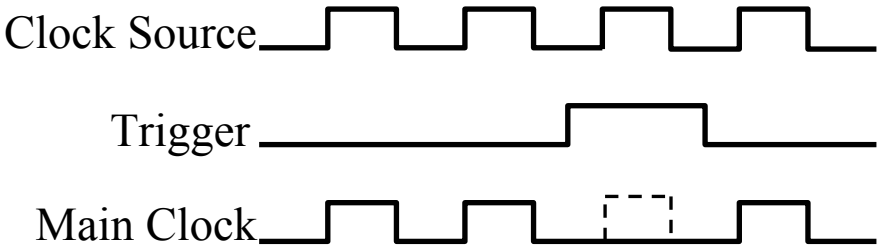


Figure 2

Clock Skewing Example

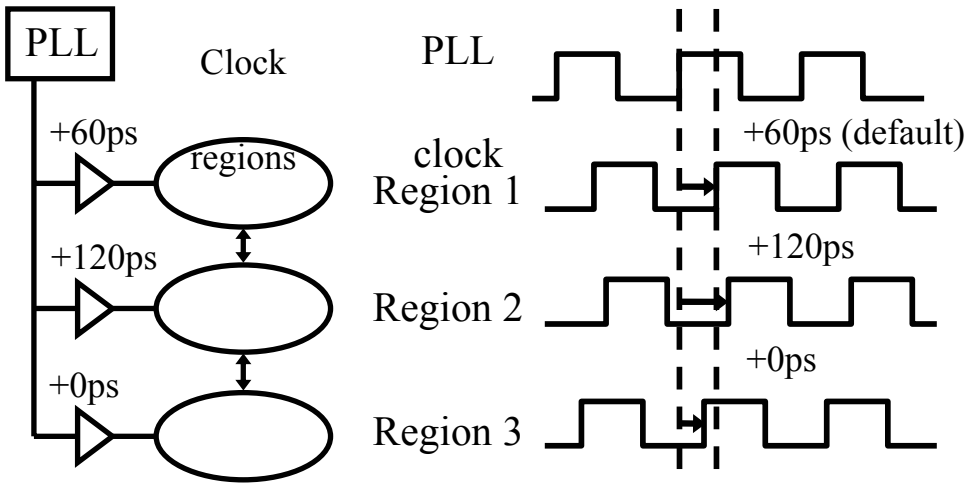


Figure 3